**Amendments to the claims,**
**Listing of all claims pursuant to 37 CFR 1.121(c)**

*This listing of claims will replace all prior versions, and listings, of claims in the application:*

1. (Currently amended) A method for refactoring a plurality of interdependent software modules that reside in separate projects, the method comprising:

in response to a change that affects a particular symbol of a software module that resides in a first project, refactoring the software module of the first project to propagate the change to all instances of the particular symbol in the software module;

during the refactoring of the software module of the first project at a given point in time, recording meta data about the refactoring that is required to effect the change; and

at a subsequent point in time, automatically propagating the change to a dependent software module residing in a second project, by refactoring the dependent software module based on the recorded meta data about the refactoring that occurred to the software module of the first project.

2. (Original) The method of claim 1, wherein the dependent software module is refactored with assistance of a symbol table used for resolving symbol references, and wherein said automatically propagating step includes:

copying symbol information about the particular symbol used for the software module of the first project into the symbol table used for refactoring the dependent software module.

3. (Original) The method of claim 2, further comprising:

removing symbol information about the particular symbol used for the software module of the first project from the symbol table used for refactoring the dependent software module after the refactoring of the dependent software module.

4. (Original) The method of claim 2, wherein said copying symbol information step includes copying symbol information into a compiler symbol table used for

refactoring the dependent software module.

5. (Original) The method of claim 2, wherein said copying symbol information step includes creating source code based on the recorded meta data.

6. (Original) The method of claim 5, further comprising:

parsing the source code so as to indirectly inject symbol information into the symbol table used for refactoring the dependent software module.

7. (Original) The method of claim 5, wherein said copying symbol information step includes indirectly injecting symbol information for a class into the symbol table by parsing the source code and directly injecting symbol information for members of the class into the symbol table.

8. (Original) The method of claim 5, further comprising:

deleting the source code after refactoring the dependent software module.

9. (Original) The method of claim 1, wherein the refactoring of the software module of the first project comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

10. (Original) The method of claim 1, wherein said automatically propagating step includes propagating changes to the dependent software module so as to maintain compatibility with the software module of the first project.

11. (Original) The method of claim 1, wherein said recording step includes recording meta data before and after refactoring of the software module of the first project.

12. (Original) The method of claim 1, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

13. (Original) The method of claim 1, wherein the software module of the first project is in a first programming language and the dependent software module of the second project is in a second programming language.

14. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

15. (Original) A downloadable set of processor-executable instructions for performing the method of claim 1.

16. (Currently amended) A system for automatically applying a refactoring to a second software module based on a refactoring of a first software module, the system comprising:

a computer having a processor and memory;

a recording module for recording information about changes made to the first software module in a first project during a refactoring of the first software module at a given point in time;

an injector module for copying symbol information about at least one symbol of the first software module into a symbol table for the second software module in a second project; and

a refactoring module for automatically applying a refactoring to the second software module at a subsequent point in time using said symbol table for the second software module and the recorded information about changes made to the first software module.

17. (Original) The system of claim 16, wherein the second software module is dependent upon the first software module.

18. (Original) The system of claim 17, wherein the refactoring module applies changes to the second software module so as to maintain compatibility of the second

software module with the first software module.

19. (Original) The system of claim 16, wherein the refactoring of the first software module comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

20. (Original) The system of claim 16, wherein the injector module creates source code based on the first software module and the recorded information.

21. (Original) The system of claim 20, wherein said injector module decompiles at least a portion of the first software module for creating source code.

22. (Original) The system of claim 20, wherein the injector module parses the source code so as to indirectly inject symbol information into said symbol table.

23. (Original) The system of claim 16, wherein the injector module copies symbol information into a compiler symbol table used by the refactoring module in the refactoring of the second software module.

24. (Original) The system of claim 16, wherein the first software module comprises a library.

25. (Original) The system of claim 24, wherein the second software module comprises an application using the library.

26. (Original) The system of claim 16, wherein said recording module records information before and after refactoring of the first software module.

27. (Original) The system of claim 16, wherein said recording module records information in Extensible Markup Language (XML) format.

28. (Original) The system of claim 16, wherein the first software module runs on a first machine and the second software module runs on a second machine.

29. (Canceled)

30. (Original) The system of claim 16, wherein the first software module is in a first programming language and the second software module is in a second programming language.

31. (Currently amended) A method for asynchronous refactoring of a plurality of interdependent software programs, the method comprising:

refactoring a first software program residing in a first project so as to change symbols of the first software program;

recording information about changes made to symbols of the first software program during the refactoring of the first software program; and

subsequently, applying the refactoring to a second software program in a second project which is dependent upon the first software program by automatically propagating changes to symbols of the second software program based on said recorded information.

32. (Original) The method of claim 31, wherein the second software program is refactored with assistance of a symbol table used for resolving symbol references, and wherein said applying step includes:

copying information about at least one symbol used in the first software program into the symbol table used for refactoring the second software program.

33. (Original) The method of claim 32, further comprising:

removing said information about at least one symbol used in the first software program from the symbol table after applying the refactoring of the second software program.

34. (Original) The method of claim 32, wherein said symbol table comprises a

compiler symbol table.

35. (Original) The method of claim 32, wherein said copying information step includes creating source code based on the recorded information and parsing the source code so as to indirectly inject symbol information into the symbol table used for refactoring the second software program.

36. (Original) The method of claim 35, wherein said copying information step includes indirectly injecting symbol information for a class into the symbol table by parsing the source code and directly injecting symbol information for members of the class into the symbol table.

37. (Original) The method of claim 32, wherein said applying step includes applying the refactoring with assistance of a compiler for identifying and changing symbols of the second software program.

38. (Original) The method of claim 31, wherein the refactoring of the first software program comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

39. (Original) The method of claim 31, wherein the first software program comprises a library.

40. (Original) The method of claim 39, wherein the second software program comprises an application using the library.

41. (Original) The method of claim 31, wherein said recording step includes recording information before and after refactoring of the first software program.

42. (Original) The method of claim 31, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

43. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 31.

44. (Original) A downloadable set of processor-executable instructions for performing the method of claim 31.

45. (Currently amended) A method for applying a refactoring to a plurality of software modules, the method comprising:

recording information about changes made to a first software module in a first project during a refactoring of the first software module;

subsequently, refactoring a second software module in a second project by performing substeps of:

creating at least one symbol table entry based upon the recorded information about changes made to the first software module;

injecting said at least one symbol table entry into a symbol table for a second software module; and

refactoring the second software module using said symbol table and the recorded information about changes made to the first software module.

46. (Original) The method of claim 45, wherein the second software module is dependent upon the first software module.

47. (Original) The method of claim 46, wherein said refactoring the second software module step includes applying changes to the second software module so as to maintain compatibility with the first software module.

48. (Original) The method of claim 45, wherein the refactoring of the first software module comprises a selected one of renaming a package, changing a method signature, renaming a method, renaming a field, and renaming a class.

49. (Currently amended) The method of claim 45, wherein said creating substep includes creating source code based on the first software module and the recorded information.

50. (Currently amended) The method of claim 49, wherein said substep of creating source code includes decompiling at least a portion of the first software module.

51. (Currently amended) The method of claim 49, wherein said injecting substep includes parsing the source code so as to indirectly inject a symbol table entry into said symbol table.

52. (Currently amended) The method of claim 51, wherein said injecting substep includes indirectly injecting symbol table entries for a class into the symbol table by parsing the source code and directly injecting symbol table entries for members of the class into the symbol table.

53. (Original) The method of claim 45, further comprising:

removing said at least one symbol table entry from said symbol table after applying the refactoring to the second software module.

54. (Original) The method of claim 45, wherein said symbol table comprises a compiler symbol table.

55. (Original) The method of claim 45, wherein said recording step includes recording information before and after refactoring of the first software module.

56. (Original) The method of claim 45, wherein said recording step includes recording information in Extensible Markup Language (XML) format.

57. (Original) The method of claim 45, wherein the first software module runs on a first machine and the second software module runs on a second machine.

58. (Original) The method of claim 45, wherein the first software module is in a first programming language and the second software module is in a second programming language.

59. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 45.

60. (Original) A downloadable set of processor-executable instructions for performing the method of claim 45.